A Knowledge Network Toolkit for Autonomic Componentware

Matthias Baumgarten[1], Nicola Bicocchi[2], Kevin Curran[1], Kieran Greer[1], Rico Kusber[2], Marco Mamei[3], Maurice Mulvenna[1], Chris Nugent[1], Franco Zambonelli[3]

[1]University of Ulster, School of Computing and Mathematics, Newtownabbey, BT37 0QB, United Kingdom
Email: {m.baumgarten, md.mulvenna, krc.greer, cd.nugent, kj.curran}@ulster.ac.uk
http://www.ulster.ac.uk
[2]DISMI - Università di Modena e Reggio Emilia, 42110 Reggio Emilia, Italy
Email: {bicocchi.nicola, marco.mamei, franco.zambonelli}@unimore.it
http://www.unimo.it
[3]ComTec - University of Kassel, Kassel, Germany
Email: rico.kusber@comtec.eecs.uni-kassel.de
www.comtec.eecs.uni-kassel.de

ABSTRACT

With the dawn of smart world infrastructures on a global scale, the need for fully autonomous operating systems and services has never been more urgent. A key aspect for such systems is the availability of relevant contextual information so that they can autonomously configure, adapt and optimize their behavior towards changing conditions. This is known as context or situation awareness, which is of fundamental importance for successful autonomic computing and services. However, acquiring relevant information from the real, virtual or operational environment is only the first step to facilitate such contextual awareness. Additional processing is required to pre-organize, correlate or simply reformat such data so that they are readily accessible as well as usable by a multitude of services. Within a distributed environment, this translates directly into a diverse network of knowledge, where individual views correspond to specific contexts and situations that are ultimately understandable yet manageable in real time. This paper, discusses a Knowledge Network (KN) approach that has been developed as part of an Autonomous Componentware toolkit, called ACE-Toolkit (Autonomic Communication Elements). The reference based KN framework provides the means for the ACE-Toolkit and its components to acquire a higher degree of contextual awareness where knowledge from various sources can be utilized efficiently at various levels of granularity. On the other hand the KN components themselves are realized as ACE's thus taking full advantage of the autonomic features offered by the toolkit.

Keywords

Autonomic Computing, Knowledge Networks, Self-Organization

1. Introduction

Smart environments are already a reality but exist often only in isolation, on a smaller scale and for a dedicated purpose, e.g. smart homes. However, latest advances in communication as well as in sensor technology will allow to (a) connect individual smart environments together; (b) drastically increase the use of sensors for various purposes and (c) monitor real as well as virtual environments in a pervasive fashion. Naturally, this will increase the amount of data to be processed but, at the same time, provide the required knowledge for services and components to acquire a higher degree of contextual awareness. Consequently, this has the potential to result in better Quality of Service with reduced costs to the service provider. Figure 1 depicts the relation between traditional and semantic environments and their usability with increased autonomic features. Undoubtedly, the usability of such systems will be increased through the provision of autonomic features. However, the complexity will also increase due to the increase in heterogeneous devices, their interactions between each other as well as the type of services that will be deployed on these environments.

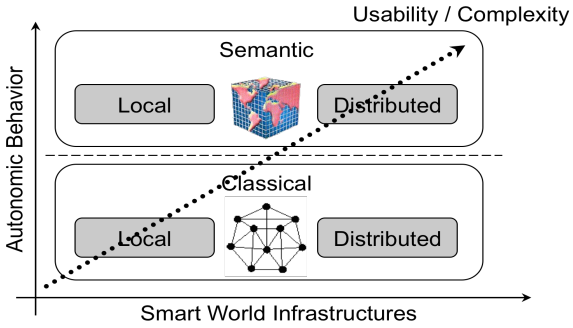

Figure 1: Usability vs. Complexity

The formulation and provision of context specific information provides the means to reduce complexity by enabling individual services to be applied to dynamic and individual rather than static and pre-defined contexts. In consequence services are tailored towards specific needs that depend on a multitude of aspects along various dimensions, including, purpose, environment, time, etc. Based on [2] context can be described as follows: "*context is any information that can be used to characterize the situation of an*

entity" and that can be considered as relevant to adapt/improve the interaction between such entities and its users. In addition to this, context-awareness can be seen as a prerequisite for services to improve quality and reliability via adaptability, e.g. by exploiting contextual information to self-monitor, self-configure, self-optimize, etc.

The reminder of this paper is organized as follows. Section 2 outlines the motivation behind knowledge networks, whereas the general framework is introduced in Section 3. The problem of knowledge querying and knowledge verification is discussed in Section 4 and Section 5 respectively. A prototype implementation is introduced in Section 6 before concluding remarks are given in Section 7.

2. Motivations

The general concept of knowledge networks is centred on four key principles: context-awareness, autonomicity, self-similarity, and semantic self-organization. While the principle of context - awareness is at the very core of knowledge networks, the other three are equally important to the overall architecture. In particular, autonomic features will play an important role if knowledge networks would be applied on a global oriented environment, such as large-scale smart environments. As far as the principle of self-similarity is concerned, we consider that individual components of a knowledge network share the same interface whether they relate to direct knowledge sources or to more structured, already pre-organised "knowledge containers". In other words, the access as well as the organisation of individual knowledge is performed by the same mechanism independent of the type, size or location of the knowledge. We consider the above properties as very important for representing, composing and evolving distributed knowledge in a robust and flexible way and most importantly to provide a light-weight framework. In particular, it will be interesting to explore how to structure a host of knowledge sources into scale-free, fully distributed networks of knowledge, so as to reflect the context they reflect and at the same time to support robust adaptive evolution thereof. Moreover, the study of such network structures could be of use to support the scalability of such structures and, possibly even more important, to better support scale-free composability as well as self-similar multi-level perception of knowledge at different scales of observation. In turn, applications and services can take direct advantage of such knowledge constructs, which may even organised not only semantically bit also along other dimension such as spatial, temporal or even along additional application-specific dimensions. This obviously requires more advanced knowledge structures that dynamically incorporate required dimensions.

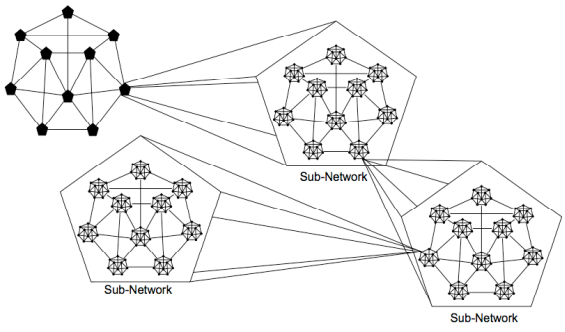

Figure 2: Network of Networks

As far as semantic self-organization is concerned, it is envisioned that knowledge sources are self-descriptive and as such provide relevant information that can be used directly or indirectly (e.g. via some sort of validation, conversion, translation, etc.) for organisational purposes. Generic self-organization algorithms may then be employed to discover and enact relations among these initially uncorrelated pieces of knowledge. The relations established could then be grouped into dedicated knowledge containers, which reflect knowledge clusters that are relevant to the context of the container they belong to. From the emerging network of such relations, it may then be possible to acquire new knowledge about facts and situations, which in turn can be published again into the scope of the KN, thus acquiring an introspective view about the knowledge embraced. As shown in Figure 2, dedicated networks of networks can be constructed in which knowledge is stored at the leaves and organised via its branches.

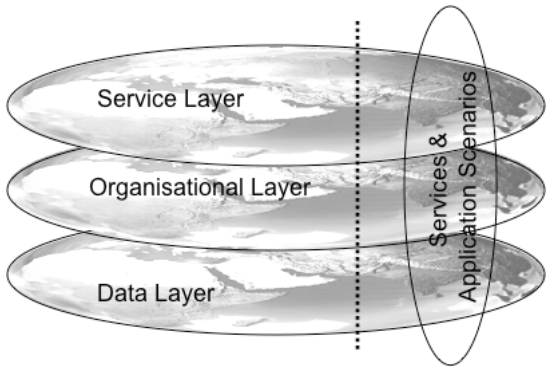

Figure 3: Conceptual Layers of Knowledge Networks

Simplified, a network of knowledge can be seen as a reference-based structured collection of knowledge atoms, containing contextual information at different levels of granularity, and built in order to promote the contextual-awareness of other systems and services. Unlike in the "knowledge plane" approach [3], we do not consider KN's as a heavyweight control plane for services, where to embed logics of application control and management. KN's are designed to stay light-weight by only referencing other nodes but never duplicating their content. As such KN's only embed logic of information management, and rather simple logic for their internal unsupervised maintenance. As shown in Figure 3, KN's are the organizational layer, which connects the conceptually lower oriented data layer with the higher oriented service layer providing access to structured knowledge rather than raw data.

3. Knowledge Network Framework

As mentioned already, the objective of the knowledge network approach is to organize data in a way that makes it easier for a user to retrieve specific context and situation aware knowledge rather than raw data. The two base components that are utilized to construct knowledge networks are knowledge atoms, KA, and knowledge containers, KC. While the former facilitates generic access to the data the latter is solely concerned with the organization thereof. In theory, the knowledge network will be hierarchical, grouping KA's based on their semantic meaning or other available dimensions. Each KC may then contain references to the set of KA's it groups together. Alternatively, a container may also reference other containers, thus creating a hierarchical or network like architecture.

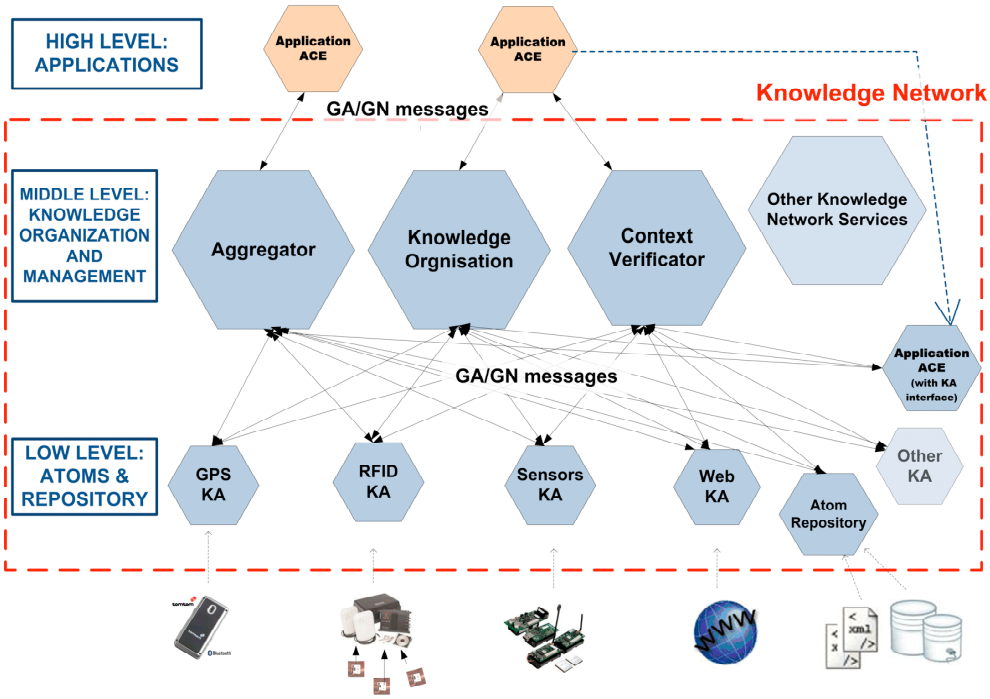

Figure 4: Knowledge Network Architecture - Schematic

This architecture has been depicted in **Figure 4**, which also highlights the three conceptual layers a knowledge network operates in as shown in Figure 3. In order to organize the knowledge based on their semantic description, three different methodologies can be used which are described as follows:

- **Bottom-Up through self-descriptive Knowledge Atoms:** Constructing relations entirely form the semantics provided by the KA's itself is the most efficient way as relevant relations can be generated upon knowledge registration. The obvious advantage of this method is that the resulting hierarchical or network like knowledge structures can be generated autonomously without any interference from the user. However, if the semantics provided by the data layer is incorrect or purposely falsified then the resulting knowledge structures are also incorrect.
- **Top-Bottom through existing ontology's:** For instance, if an ontology is known beforehand for a particular domain, then this ontology may be modelled via knowledge containers and declared static in a way that it may not be altered by internal self-organisation mechanisms. The semantics of KA's may then be validated against these pre-generated knowledge structures and may only be included if deemed correct. While this renders the organisational space to be static it also ensures that knowledge can only be mapped into the existing structures thus avoiding the generation false relations. In addition, a user or application can directly specify the type and structure of the knowledge of interest. This allows for the generation of purpose build knowledge structures serving specific needs as well as specific contexts.
- **Mixed Construction and validation:** In practice, individual relations or full scale ontology's cannot always be provided for any scenario. Furthermore, different ontology's may be required for different applications. Thus, generating KN's dynamically via self-descriptive knowledge atoms is always preferable. Nevertheless, the resulting knowledge structures should be validated wherever possible in order to provide a continuing quality of service. Such a service can be performed at three different levels; (a) when creating a distinct relation, (b) as a background service that constantly compares existing and validated knowledge with the structures created and (c) at application level, where the knowledge consumer provides the ontology's individual knowledge sources have to conform too in order to be included.

```
<Semantics>
        <Keyword>Weather.Wind Force</Keyword>
        <Keyword>Europe.UK.Belfast</Keyword>
        <Keyword>GPS[Standardised GPS Information]</Keyword>
        <Keyword>metoffice.gov.uk</Keyword>
</Semantics>
```

Figure 5: Semantic Description - Example

The semantics describing a knowledge source should incorporate individual keywords but also any relations among them. Thus, as depicted in Figure 5, a hierarchical namespace may be used that employs some form of delimiter to separate individual keywords yet maintaining their relation. Alternatively semantics may be represented in RDF thus allowing for more complex representations as well as a better understanding about the state of the knowledge resource. Constructing such relations from the bottom-up (that is from the data level) requires that that each atom is sufficiently self-descriptive to describe not only itself but also its relation to other concepts. For instance, a knowledge source representing the wind force @ location Belfast may provide a set of keywords as shown in Figure 5. Thus providing all necessary information to either link or generate respective ontology's as shown in Figure 6. Note, that such an ontology is always shared, which means that no part of any given construct is created or used

only for a single object but always available for all objects that are within the same organisational space. Thus, as shown, the ontology depicted for the example atom depicted may contain other concepts, e.g. the Weather.Wind.Direction created previously. Alternatively, KC's may be added directly as visualised through the Europe.Italy object, which is not yet linked to any knowledge source.

So far, the above has only dealt with static semantic concepts such as the location (provided semantically), the purpose and the domain of the knowledge to be mapped. However, other more dynamic concepts also have to be dealt with. Probably the best example of this is GPS data that provide the geographical location of a resource. When static they may be translated into a more meaningful and human understandable semantic representations such as addresses, town names etc. On the other hand when used for movable resources (e.g. mobile phones and RFID tags) such mapping is not always possible or desired. On the contrarily, GPS data can be used directly for geographical mapping purposes, distance calculations, clustering etc. For this to be used efficiently within knowledge networks such data should be linked to more active knowledge containers that utilise distinct algorithms that provide purpose based organisation. This can be achieved dynamically by overloading the standard algorithm with any purpose built mechanism desired. Nevertheless, one should consider the fact that self-organisation can never be achieved spontaneously so that for highly volatile concepts successful organisation may not be possible due to the simple fact that it is already obsolete at the time of organisation. Finally, considering that KA's as well as KC's are fully independent computational entities, they may be hosted locally. For instance, a sensor could be referenced directly without any additional in-memory objects, whereas network nodes that have access to any number of resources, etc could be the host for individual KC's. This fully distributed character would guarantee the lightweight knowledge structure desired as well as scalability as each knowledge object operates independently and as such can be independently accessed.

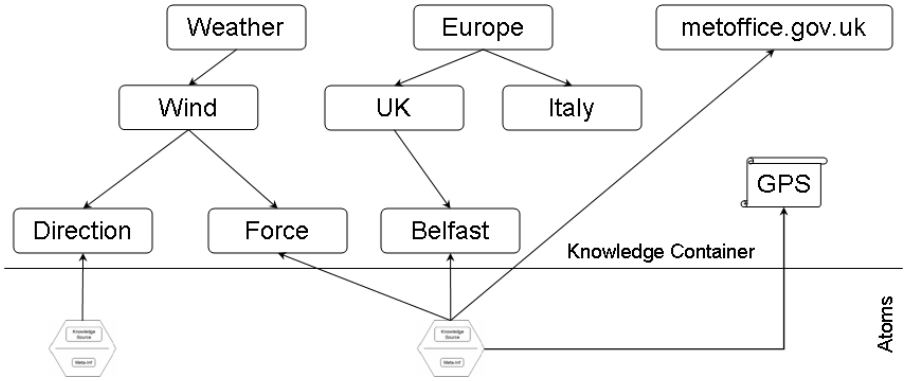

Figure 6: Semantic Based Knowledge Organization

4. Knowledge Querying

Following the example depicted in the previous Section, a KN may comprise any number of KA's that are organized based on various semantic concepts, which are in turn represented via individual KC's. Assuming that this structure exists and is readily available, the question arises of how it can be efficiently queried for knowledge. There are a number of query languages that can be used to query XML, which is used to represent the knowledge that is published by individual KA's. One possibility is to have two different phases to the querying process. The first phase is the search through the network to find the relevant knowledge sources. The second phase is to actually query the sources itself for specific constraints. This is appealing due to the fact that the search and the query process can be separated into individual tasks, which also means that different languages can be utilized for each process. The main drawback to this approach is that it is more centralized than distributed and so goes against an autonomic approach. An alternative approach would be to perform the query search and actual query execution simultaneously. That is that each KA provides its own query interface and as such will be queried individually. Here it is important that the query interface allows for heterogeneous query engines with respect to the query language used so that different knowledge sources can be queried. If we have a complex XML structures, then the query engine might be something like XQuery or Xcerpt. If we have a simple sensor, then an RDF based query language could be utilized directly. Thus each KA will receive a query request from a knowledge requester, convert it into a format suitable for its own query engine, execute it and then return the result. While the nature of this approach is more distributed and also less intensive with respect to network traffic it also requires more "intelligence" form the knowledge atoms itself, which is against the lightweight approach desired. An additional level of complexity is introduced when KA's are queried in dependence of other KA's. In this case a more centralized query mechanisms may be preferable. If using a select-from-where statement, the centralized and distributed approaches can be partially combined. The 'where' comparisons can be evaluated locally at KA's that compare knowledge to exact values, e.g. 'Temperature greater than 21 degrees', whereas more complex evaluations are perfumed via a central query engine.

The following depicts a scenario that indicates what the combination of both approaches would allow: Say we have a number of temperature sensors distributed over e.g. Belfast and we have a number of XML documents with knowledge on the buildings in Belfast and the people that work in them. A possible query could look like "Retrieve the temperature from buildings where people work who wear ties and eat cornflakes" or in SQL like format

```
select Weather.Temperature, from Weather where
Weather.Building in (select Buildings.Name from Buildings
where Buildings.People in (select People.Name from People
```

where People.Wear equals Ties and People.Eat equals Cornflakes))

If we enforce a nesting for processing then this could be done in individual stages. We first query the people, then retrieve the buildings and query that and then finally query the temperature sensors. While the most inner (equals) operations could be evaluated locally, that is in a distributed fashion, other operation such as the gathering of all building names could be done in a centralized fashion. Whatever way the querying is performed, the important aspect here is that the query is capable of reflecting a specific context or situation, which in turn is answered through the knowledge of the network. While XML processing would be desirable for a complete system in order to guarantee interoperability, it is not absolutely necessary. The select-from-where statement that queries for simple values should suffice for most sensor based knowledge sources, which again fosters the lightweightness of the knowledge network.

The above discussion has mainly concentrated on aspects that are concerned with the querying of KA's rather than with the efficient identification of KA's that are relevant to fulfill a specific query. This aspect, however, is important when considering that a KN may comprise very large numbers of KA's. Hence, the optimization of the search process is vital for the response time of the query process. Simplified, when querying a network then the query process follows specific routes through the network in order to find relevant KA's that satisfy the query. If the user is satisfied with the query result, then the knowledge network itself may be updated to reflect these pathways, which reflect successful knowledge retrieval. In this way, the network could be modified in an experience-based way that is based on its use rather than solely on its content. A relatively simple way to do this is the following: We note the nodes visited to answer a query and construct links between nodes commonly associated together. So if a user submits a query, the first part is constructed and a path through the network is navigated. The sources process the query and send the reply back through the network. The nodes that receive the reply, note that they have been used in processing this query. If the user then confirms that the query result is acceptable, these nodes then permanently update an internal list that stores references to other nodes that have been used to answer the query or any part thereof. This aspect has been further discussed in [5].

5. Knowledge Verification

Equipping applications with the ability to adapt to different contexts and situations requires data about the applications' environment. Data that is required to construct high level contextual information has to be sensed, measured, or derived before it can be used as an input for a context aware application. In the flow of this process errors can occur at several points. Defective hardware, misinterpreted data out of date data, or unforeseen circumstances may lead to invalid contexts, which in turn could lead to further problems for the service that uses such false contexts. For this reason it is necessary to include mechanism that are capable of verifying contextual data before they are used. Contextual data can be distinguished into low-level context and high-level context, where low-level context comprises data directly derived from sensors and high-level context is information that is inferred from low-level context. For example, the longitude and latitude of a GPS data set is low-level, whereas the interpretation of this data to a specific address is considered as high-level context. Two ways of context verification are possible with in the context of knowledge networks. Firstly, verification via reference patterns in which a context in question is compared to a number of references patters that has been previously collected and rated. Secondly, he correlation of individual contextual date with reference data that is also available via the network. For instance, a temperature sensor could be validated via neighboring sensors. While the former is more suitable for high-level context verification the latter is favored for the verification of low-level context.

6. Implementation

In order to explore and validate the outlined knowledge network approach a prototype implementation has been developed as part of the CASCADAS[1] project. The core components of knowledge networks (knowledge atoms and knowledge containers) have been realized as ACE's (Autonomic Communication Element) via the ACE toolkit [4], which provides a platform for the development of autonomic services. As depicted in Figure 7, knowledge sources have the potential to self-register into the "ACE Universe", which comprises any number of Knowledge containers that collectively reflect the knowledge network. The fact that individual objects (KA's, KC's or any other ACE object) may exist on different computational resources is irrelevant as this aspect is hidden form the user.

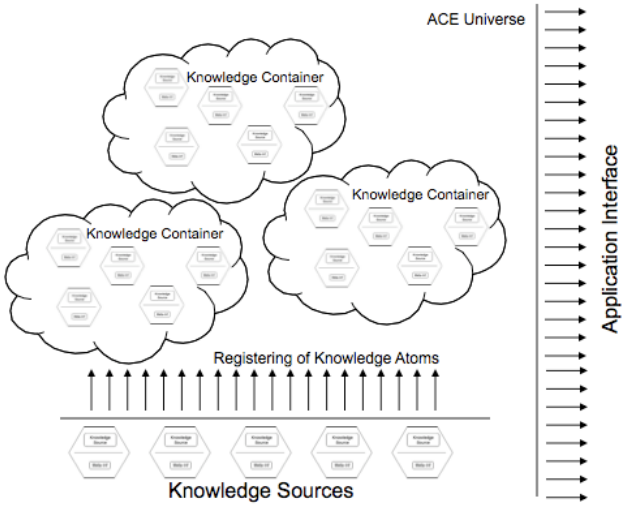

Figure 7: Knowledge Network Realization

Once registered, KA's are organized based on their semantic description resulting in dynamic constructs that can be maintained and queried efficiently. Applications may query the network or once known, access the knowledge sources directly. A visualisation thereof is shown in Figure 8, which shown a small network where green nodes refer to distinct knowledge containers and white nodes to knowledge atoms. A tree like visualisation thereof is also shown on the left side.

7. Conclusions & Future Work

Self-organizing networks of knowledge is an important aspect for future autonomic services as well as for large-scale smart environments. The concept of knowledge networks may have the potential to bride the gap between these two areas so that services

[1] http://www.cascadas-project.org

can become more contextual aware by profiting form the data that is made available through smart world infrastructures. On the other hand smart environments would also benefit through highly adaptive services that operate in a context aware way.

Although promising, the developed prototype still has to be evaluated at a larger scale including great numbers of heterogeneous knowledge sources as well as different types of services that have different contextual requirements. Another aspects to be researched include the utilization of more diverse self-organizational algorithms that go beyond static semantic concepts. Adding predictive capabilities to the network itself is another challenging research direction. For instance, a given context reflected by the network may be observed and analyzed over time. This could allow for the prediction of individual aspects of this context or of the context as a whole, which in turn would allow for the detection of invalid or dangerous situations before they actually occur. Similar to this, flexible reasoning mechanisms would be highly desired in order correlate unrelated knowledge sources with each other which could then be utilized for e.g. knowledge verification purposes. Bio-inspired mechanisms, which incorporate concepts such as stigmergy, swarm intelligence and heartbeat signals offer additional perspectives that are relevant for the optimization and adaptation of knowledge structures that are complex in nature, highly distributed and often unrelated with each other or with the domain /situation they are used for. In theory, they have the potential to enable distinct optimization without the requirement for an intrinsic stimulus, which allows for independent and truly distributed optimisation procedures, which could help to make such systems not only more robust and reliable but also more efficient.

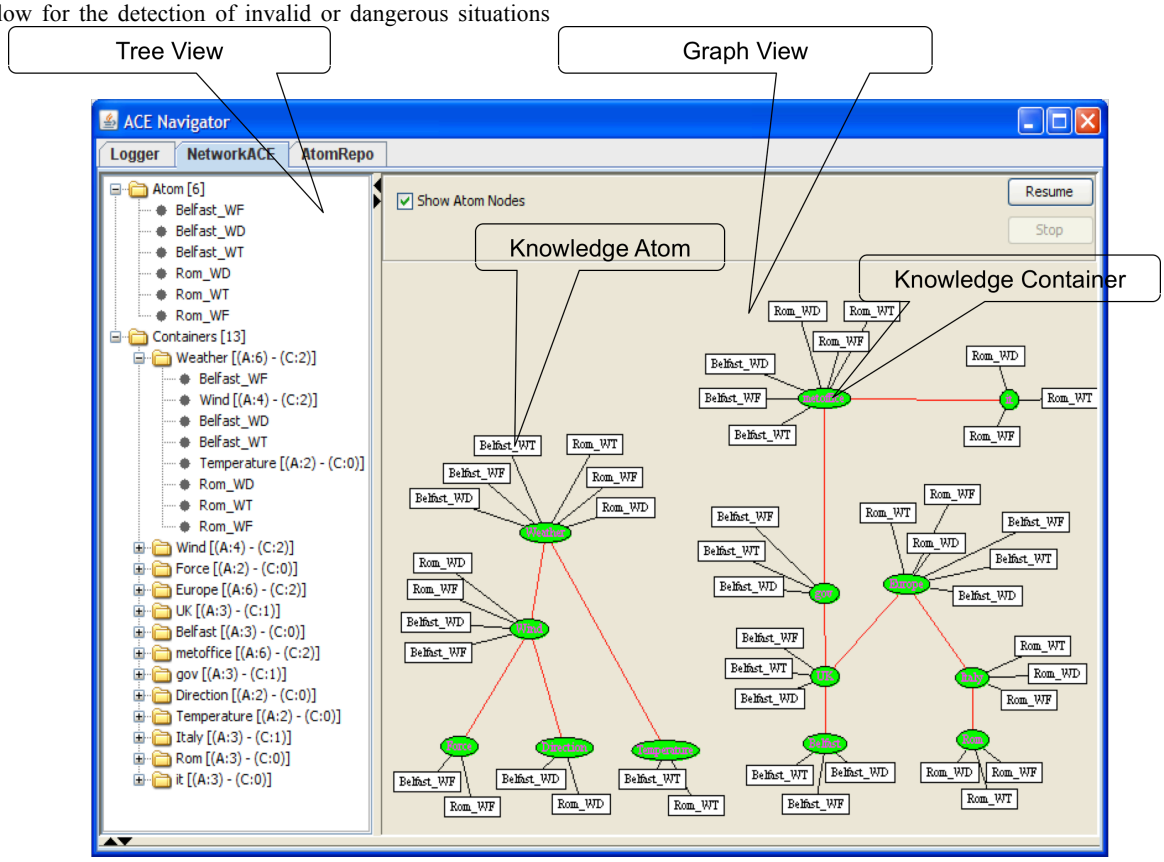

Figure 8: Knowledge Network Visualization

8. Acknowledgments:

Work supported by the project CASCADAS (IST-027807), FET Program of the European Commission. (http://cascadas-project.org)

9. References

[1] M. Baumgarten N. Bicocchi, M. Mulvenna, F. Zambonelli, "Self-organizing Knowledge Networks for Smart World Infrastructures", International Conference on Self-organization in Multiagent Systems, Erfurt (D), 2006.

[2] T. Buchholz, A. Kupper, S. Schiffers, "Quality of Context Information: What is it is and Why We Need It", 10th HP-OVUA Workshop, Geneva (CH), July 2003.

[3] D. Clark, C. Partridge, C. Ramming, J. Wroclawski, "A Knowledge Plane for the Internet", 2003 ACM SIGCOMM Conference, Karlsruhe (D), 2003.

[4] https://sourceforge.net/projects/acetoolkit/

[5] Kieran Greer, Matthias Baumgarten, Chris Nugent, Maurice Mulvenna and Kevin Curran, (2008). Autonomous Querying for Knowledge Networks, The 5th International Conference on Autonomic and Trusted Computing (ATC-08), June 23 - 25, Oslo, Norway